bLoyal

POS Application

Integration Guide

Last Updated: February 27, 2014

"Helping you build loyal customers" ***TM***

Table of Contents:

1. Introduction 2
2. Application Connector Certification Levels 2
3. Integration Overview 3
4. Loyalty Engine – Promotions and Awards 5
4.1 Inflow Promotions and Redemptions 5
4.2 Tender Based Promotions and Redemptions 7
4.3 Recording Visit Transactions 9
4.4 Customer and Subscriber Group Signup 9
5. POS Snippets – Customer Signup and Orders 11
5.1 Supported POS Snippets 12
6. Channel Hub™ – Backend Data Integration 14
6.1 Protocol – Register() and Checkin() 14
6.2 Synchronizing Entities 15
6.3 Supported Entities 17
7. Revision History 18

1. Introduction

bLoyal is a cloud based customer loyalty and order processing platform that works across channels and devices. Our goal is to make it easy for you to add bLoyal services to your POS application.

This integration guide describes how to integrate bLoyal with a POS application and describes the different levels of certification. The guide is intended to provide everything you need to complete your integration though you may find the web service reference guide and the web service SDK samples useful in completing your integration. Both can be found in the bLoyal API Support Center Forums.

2. Application Connector Certification Levels

There are three levels of integration for a POS application.

- **bLoyal Enabled** – A bLoyal enabled integration is simply a backend integration where the POS system is sending Sales Transactions, Products, and Customers to bLoyal. A bLoyal Enabled integration provides a one-way data synchronization from the POS system to bLoyal where bLoyal accrues benefits and generate awards that can be redeemed via any one of the “disconnected” redemption scenarios provided by bLoyal such as the **My Mobile Loyalty™** application or the **Scan-To-Redeem** POS Snippet.

- **bLoyal Certified** – A bLoyal certified integration supports bLoyal promotions and award benefits inflow of the POS transaction using the CalculateSalesTransaction() web service method. A Certified integration also supports bLoyal as a gift and loyalty tender using the bLoyal Gift and Loyalty processor APIs which enable real-time redemption and tracking of discounts, offers, and coupons.

- **bLoyal Gold Certified** – A bLoyal gold certified integration builds on the certified integration to provide the bLoyal order processing engine at the POS to enable omni-channel order processing capabilities such as POS shipping orders and pickup orders.

3. Integration Overview

A POS integration involves the following three types of actions or steps.

1) **Step 1: Backend Data Integration** – Certain data needs to flow between the POS system and bLoyal. This can either be a one-way data flow between the POS system and bLoyal or a 2-way data synchronization between each system. This backend data integration is handled using the bLoyal Channel Hub web service (ChannelHub.svc)

2) **Step 2: Promotions and Award Redemption** – Promotions and loyalty rewards can be redeemed as tenders or as discounts inflow of the transaction. The promotions and loyalty engine is accessed using the Loyalty Engine web service (LoyaltyEngine.svc)

3) **Step 3: Customer Signup and Order Processing (POS Snippets) -** Customers can be signed up for loyalty programs and clubs at the POS using the POS snippets. Additionally, you can easily add omni-channel order processing to your POS system by calling the order based POS snippets. The POS Snippets are accessed via POS application branded web pages (/POS/[SnippetName])

The table below shows the integration necessary for each level of integration.

Integration Steps	API	bLoyal Certification Levels		
		Enabled	**Certified**	**GOLD Certified**
Step 1: Backend Data Integration	ChannelHub Web Service	***One-Way -> to bLoyal*** Sales Transactions → Customers → Products →	***One-Way -> to bLoyal*** Sales Transactions → ***Two-way Synchronization*** ← Customers -> ← Products → ← Departments → ← Categories → ← Discount Reason Codes →	[Same as certified] ...and... ← Inventory Transactions Product Inventory → [Optional] ← Club Membership →
Step 2: Promotions and Award Redemptions	LoyaltyEngine Web Service	***Disconnected*** • ScanToRedeem^{TM} - POS Snippet • My Mobile Loyalty^{TM} Application	***Redeem as Tender*** • CardRedeem() • CardRefund() ***Redeem as In-Flow Discount*** • CalculateSalesTransaction() • CaptureTransaction() or CaptureToken() Optional: • ValidateCoupon() • SubmitVisitTransaction()	[Same as certified]
Step 3: Customer signup and Orders	POS Snippets	n/a	***POS Snippets - Loyalty*** • POS/OfferDetails Optional: ***POS Snippets - Loyalty*** • POS/QuickSignup • POS/FindCustomer • POS/CheckBalance • POS/ViewCustomer • POS/ApplyCoupon • POS/RecordVisit	***POS Snippets - Loyalty*** • POS/QuickSignup • POS/FindCustomer • POS/ViewCustomer • POS/ApplyCoupon • POS/RecordVisit ***POS Snippets – Orders*** • POS/CreateOrder • POS/PickupOrder • POS/CreateReturn • POS/CheckInventory

Table 1.0 – POS Integration Levels

4. Loyalty Engine – Promotions and Awards

This Loyalty Engine web service allows you to add real-time promotions and loyalty award redemptions to the POS register as part of a transaction or ticket.

https://ws.bloyal.com/v3.5/LoyaltyEngine.svc

Customers can redeem their benefits as discounts on a transaction or as a tender. bLoyal supports both methods and both are required for a bLoyal certified integration.

4.1 *Inflow Promotions and Redemptions*

There are two scenarios for adding the bLoyal Loyalty Engine inflow of the transaction. One scenario is when the customer is found and set to the transaction using the customer system of the local POS system and other is when you want bLoyal to handle finding and setting the customer. Each is shown below.

Scenario 1 – Use local POS system customer database to find/create the customer prior to calling the Loyalty engine.

In this scenario, the local applications database is used as the primary customer database and the bLoyal Loyalty processing service is used to calculate benefits and accrue rewards. The customer will be automatically created in bLoyal if the customer doesn't already exist.

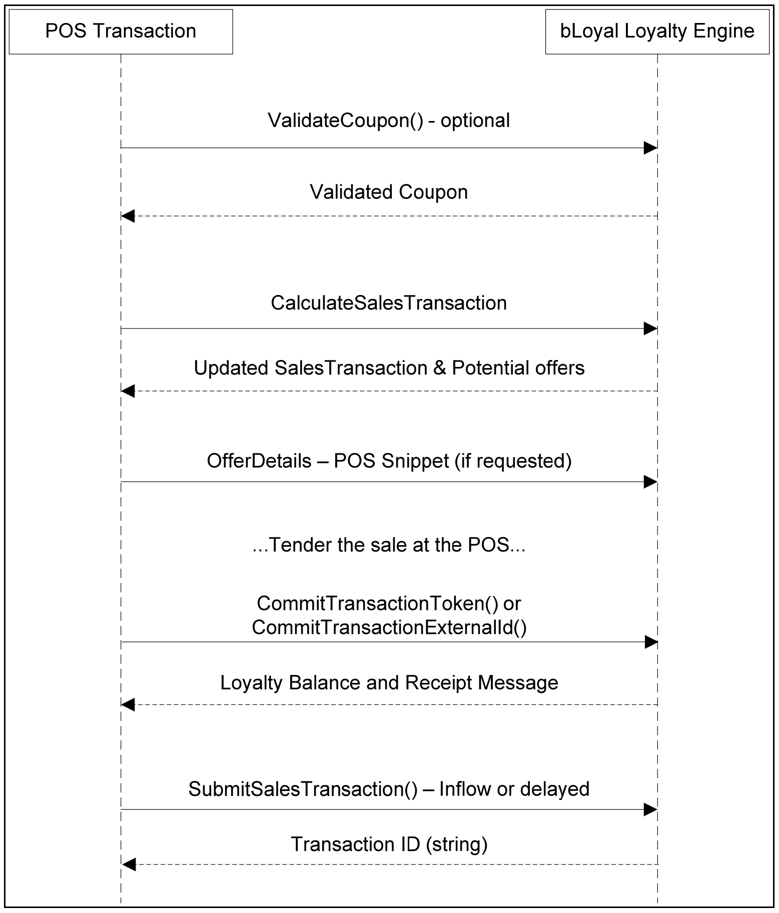
POS Transaction
bLoyal Loyalty Engine
ValidateCoupon() - optional
Validated Coupon
CalculateSalesTransaction
Updated SalesTransaction & Potential offers
OfferDetails – POS Snippet (if requested)
...Tender the sale at the POS...
CommitTransactionToken() or
CommitTransactionExternalId()
Loyalty Balance and Receipt Message
SubmitSalesTransaction() – Inflow or delayed
Transaction ID (string)

Figure 3 – Inflow with customer set in the POS system

Step 1: Assuming your POS system has a screen where the cashier can enter a coupon code, you can validate that coupon using the **ValidateCoupon**() method. This is an optional call if you want to provide immediate feedback to the cashier. The coupon will be validate regardless when you pass it in as part of the transaction to the CalculateSalesTransaciton () method. You can also call the POS/ApplyCoupon POS Snippet instead of providing your own coupon entry UI.

Step 2: Call **CalculateSalesTransaction**() - The CalculateSalesTransaction method takes a sales transaction as input (and any Coupons the cashier applied) and runs it through the bLoyal promotions and loyalty engine to determine which, if any, discounts apply.

The method will return the following information in a CalcResponse entity.

1. A modified sales transaction with any automated discounts that applied.
2. A list of potential offers the customer may be eligible for (if any). This includes an OfferToken and a text description of the offer to communicate to the customer.
3. A summary of any loyalty points or currency that will accrue or be used when the sales transaction is completed (if applicable).

The POS system should display any offers that are returned and ideally show the loyalty points that will accrue and/or be used by the transaction if either is returned.

A transaction Token will be created and returned by the CalculateSalesTransaction method. This token maintains a link between the POS sales transaction and any proposed discounts, offers, and loyalty accruals that are applicable to this Sales Transaction.
Important: If you make any changes to the Sales Transaction after calling CalculateSalesTransaction, you will need to make another call to CalculateSalesTransaction. You sould either pass in the same transaction token value generated previously or you can pass in your unique transaction number as the TransactionExternalId.

The CalculateSalesTransaction() method should be called automatically prior to tendering a sale. The POS system should also provide a way to manually to call CalculateSalesTransction() so the cashier can check at any point during the transaction. It should be called on every transaction at the POS regardless of the whether a customer is set to the transaction or not.

Step 2b: Launch the **OfferDetails POS Snippet** – Any offers that were returned in the CalculateSalesTransaction() method will have a description and an OfferToken to be used for more information about the offer. If an Offer was returned as part of the CalculateSalesTransaction method, then the POS system should notify the cashier in some way and provide a “Offer Details…” button to enable the casher to get more information about the offer and redeem the offer if the customer so chooses. When the “Offer Details…” button is selected it should launch the OfferDetails POS snippet with the OfferToken passed in on the query string.

Step 3: Call **CommitTransactionToken()** or **CommitTransactionExternalId()** – Once the sales has been successfully tendered the POS system should either call CommitTransactionToken() or CommitTransactionExternalId(). This lightweight call will capture any loyalty benefits that were redeemed on the transaction. It will also return the customers currently loyalty program balances and any receipt messages that merchant wants to show on the receipt.

Note: The CommitTransactionExternalId() method takes as input the unique transaction number (TransactionExternalId) of the POS system. The CommitTransactionToken() method takes as input the transaction Token that was generated by CalculateSalesTransaction. The transaction Token should be used when the POS system does not yet have a unique transaction number to assign to the transaction. Not all POS systems have a transaction number created when they first call CalculateSalesTransaction(). If that is the case for your POS system then you should use the transaction Token when calling any of the calls in the sequence after calling CalculateSalesTransaction(). If you do have a unique transaction number when calling CalculateSalesTransaction then you can just use your number passed in as TransactionExternalId of the bLoyal generated Token.

Step 4: Call **SubmitSalesTransaction()** – All sales transactions should be submitted to bLoyal regardless of whether a customer was assigned or a benefit was applied. This is done via the SubmitSalesTransaction() method which can be called either directly after the sale has completed or later as part of the backend data synchronizing as described in the Channel Hub section below (section 6).

Scenario 2 – Use bLoyal customer database to find/create customers prior to calling the Loyalty engine.

In this scenario, the bLoyal customer database is used as the primary customer database. This enables global customer viewing and editing across stores and sales channels. You can either use the Loyalty Engine web service calls for finding and creating customers or launch a POS snippet (as described in section 5).

Step 1: Launch the FindCustomer POS snippet to find the customer(s) in the bLoyal system…or use QuickSignup POS snippet to quickly find and/or create a customer.

The remaining of the steps are the same as Scenario 1.

4.2 Tender Based Promotions and Redemptions

Promotions and awards can also be redeemed by the bLoyal loyalty engine as a payment tender using the following tender types: Gift and eGift Cards, Loyalty Currency, and Certificates (aka Vouchers). Adding bLoyal the gift and loyalty processing to your application is very similar to any other gift card processor.

There is a single set of API methods you need to call regardless of the type of tender. You pass in a TenderCode and CardNumber and the Loyalty Engine will determine whether it's a Gift Card, eGift Card, Loyalty Currency, or Certificate.

The complete list of card processing calls is below.

- **GetCardBalance()** – Returns the current and available balance of the Gift card, eGift card, Loyalty card, or Certificate.
- **CardAuthorize()** – Authorizes a charge to the card and puts a hold on the card
- **CardRelease()** – Releases an authorization/hold that was put on a card..
- **CardCapture()** – Captures a previous authorization
- **CardRedeem()** – Redeems a value to a card (the same as Auth/Capture).
- **CardRefund()** – Refunds an amount to a card based on a previous capture or redeem transaction.
- **CardCredit()** – Loads a card with a amount. This is used when a customer activates a gift card or loads an amount to a card.
- **CardDebit()** - Take an amount off a card.

The sequence diagram below shows how to redeem as a payment tender.

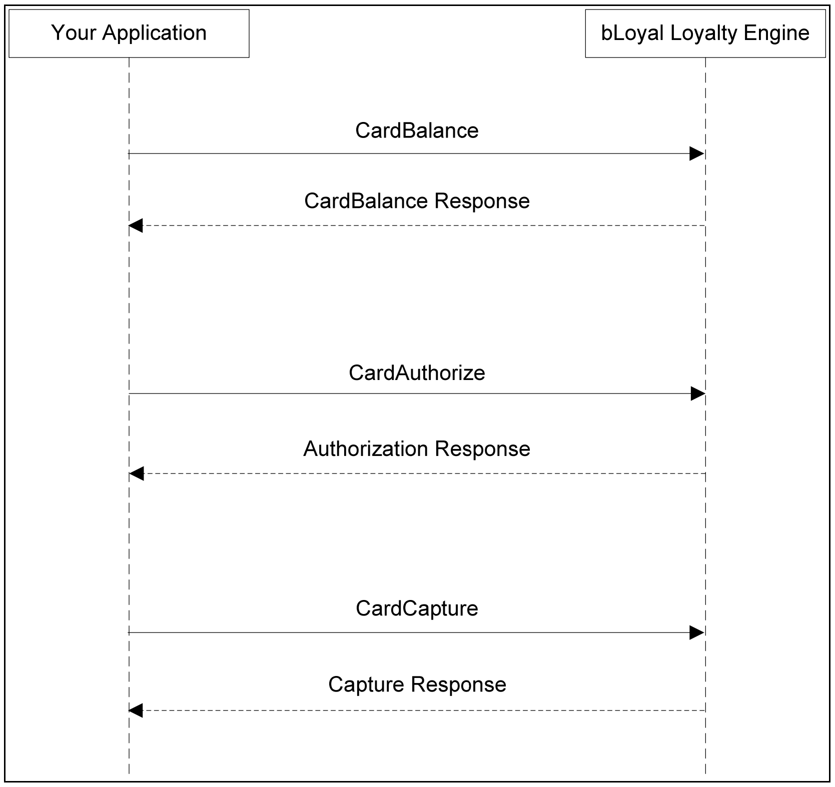

Figure 4 – Gift and Loyalty Processing as a Payment Tender

The following sequence diagram shows how to load a balance on a card.

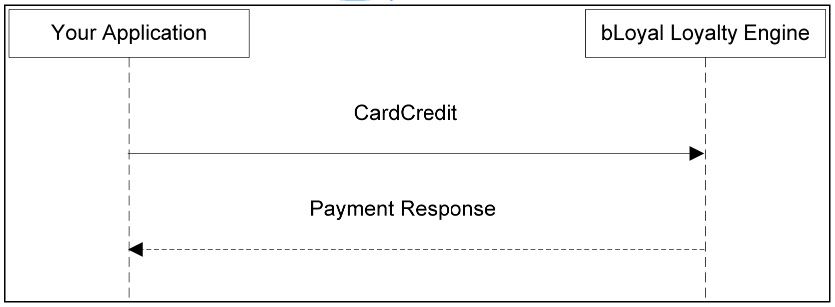

Figure 5 – Loading a balance on a gift or loyalty card.

4.3 Recording Visit Transactions

Visits can be recorded for physical or virtual (web site) visits and allow customers to accrue loyalty points and/or loyalty currency among other benefits. The sequence diagram below shows how to get the available visit types for a given Device and then record a visit transaction.

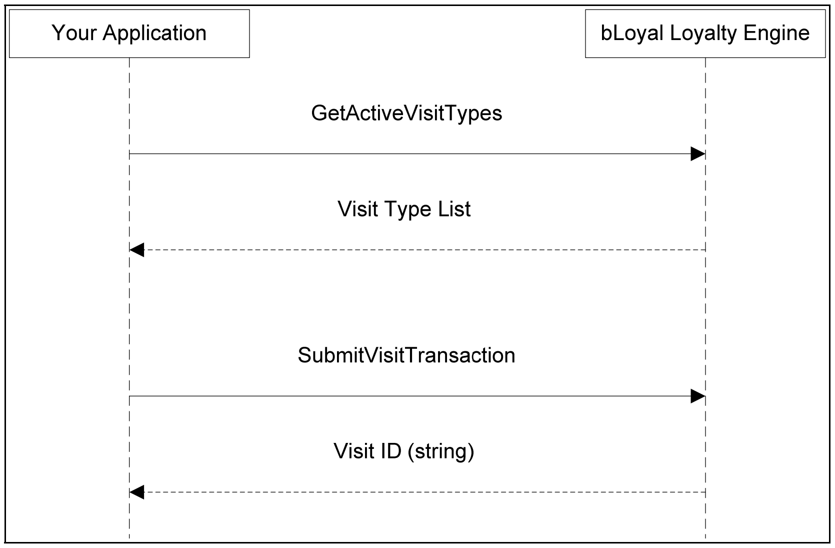

Figure 6 – Recording a Visit Transaction

- **GetActiveVisitTypes()** – Returns a list of currently active visit types that are available for the Device you are using.
- **SubmitVisitTransaction () –** Records a visit transaction for the requested visit type. The method will create a new customer in bLoyal if the customer does not exist. If no visit type is provided then a visit transaction is recorded with the default visit type assigned to the device.

4.4 Customer and Subscriber Group Signup

Capturing customer information and adding them to subscriber groups are key pieces in accruing loyalty benefits. A customer record must exist for a customer to accrue loyalty points and dollars. A Subscriber Group in bLoyal is an externally available membership list that a customer can opt-in to. Subscriber Groups are used to include customers in loyalty programs and mailing lists.

The easiest way to add customer signup capability to your POS application is using the POS Snippets (described in section 5). However, if you wish to have your own native application UI for capturing customer data and subscriber group memberships, you can do so using the Loyalty Engine web service. The sequence diagram below shows the web service calls for signing up customers.

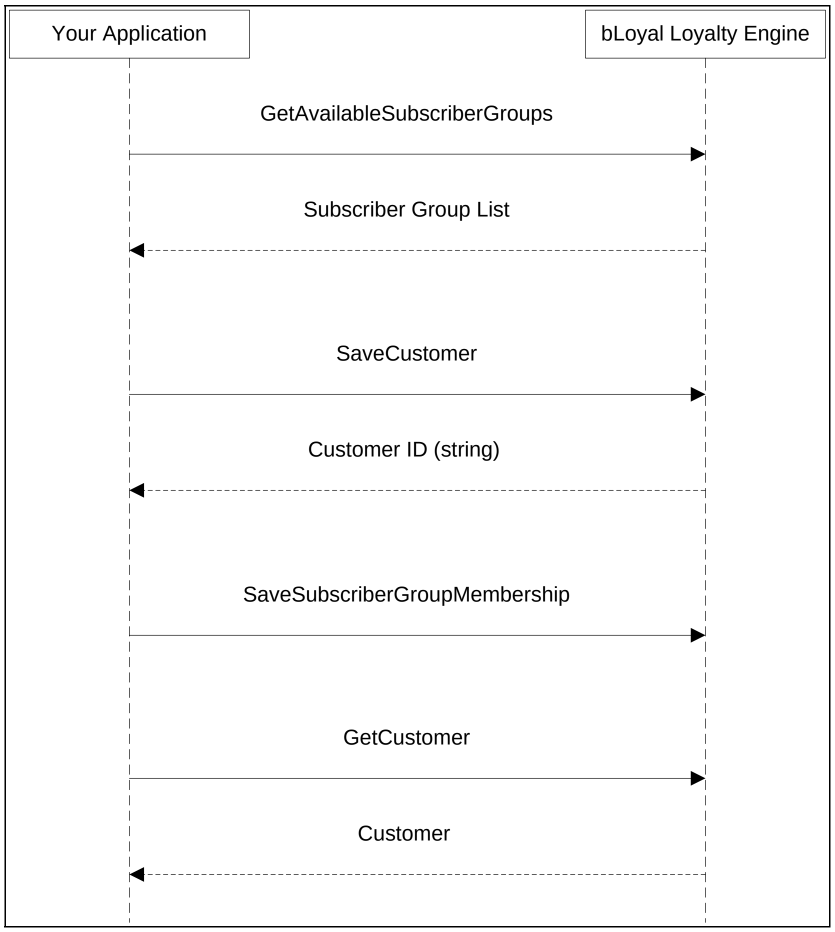

Figure 7 – Customer Signup with Subscriber Group

- **GetAvailableSubscriberGroups()** – Returns a list of subscriber groups currently available for customer signup.
- **SaveCustomer()** – Creates a new customer or applies changes to an existing customer.
- **SaveSubscriberGroupMembership()** – Create a subscriber group membership for the requested customer.
- **GetCustomer()** – Returns the core customer information.

5. POS Snippets – Customer Signup and Orders

POS Snippets are specialized bLoyal hosted web pages that provide targeted bLoyal functionality to cashiers. The POS Snippets can be branded to match your POS application and intended to be an easy way to add additional customer and order processing capability to your POS system.

POS Snippets interact with the CalculateSalesTransaction() Loyalty Engine web service method using either your unique TransactionExternalId or the bLoyal generated TransacitonToken.

Launching a POS Snippet is easy. Your POS system needs to be support having a "Customer POS Button" that can launch a browser in a framed window in a modal type dialog (meaning the POS system should wait until the user is done with the window before doing anything else).

See the sequence diagram below for how to call a POS Snippet.

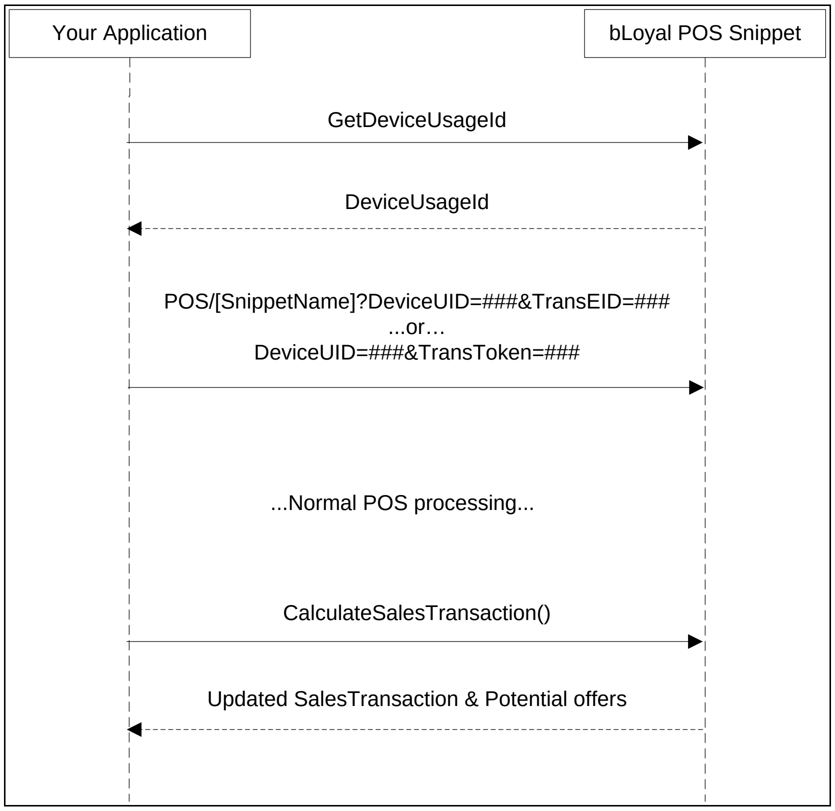
Your Application
bLoyal POS Snippet
GetDeviceUsageId
DeviceUsageId
POS/[SnippetName]?DeviceUID=###&TransEID=###
...or…
DeviceUID=###&TransToken=###
...Normal POS processing...
CalculateSalesTransaction()
Updated SalesTransaction & Potential offers

Figure 8 – Launching a POS Snippet

- **GetDeviceUsageId()** – In order to call a POS Snippet you need a device usage ID that binds snippet call to the bLoyal Device assigned to the POS Register. The GetDeviceUsageId() method simply uses the web service login to create a session key that is returned as a DeviceUsageId. This should be called prior to launching any POS Snippet URL.

- **Launch the POS/[SnippetName]** – When launching a snippet URL you need to include the DeviceUsageId by the previous step as **DeviceUID** on the query string along with either your local unique transaction number as query string parameter **TransEID** or transaction **TransToken** that was returned by the CalculateSalesTransaction() method.

5.1 Supported POS Snippets

bLoyal provides several POS Snippets. They are all used in conjunction with the CalculateSalesTransaction() method as the mechanism used to exchange sales transaction data between POS system and bLoyal. POS Snippets are configured by the bLoyal Client based on their unique configuration and requirements and are branded to match the POS application so the cashier experience is consistent.

All POS Snippets are launched in the same way by POSTing the UsageId hidden parameter and passing in either **TransEID** (the Transaction ExternalId which is your unique transaction number) or **TransToken** (the bLoyal generated transaction token) on the query string. The base URL for the snippets is: https://snippets.bloyal.com/POS

Blow is the list of POS Snippets provided by bLoyal. Ideally your POS application can support calling a POS snippet by adding a custom POS button to register.

bLoyal Enabled integrations use the following POS Snippet:

- **/ScanToRedeem** – This Scan-To-Redeem snippet provides a simple UI for the cashier to scan or lookup a customer to view and manually redeem the customer benefits. This is provided for systems that have not yet integrated with the CalculateSalesTransaction() method.

bLoyal Certified integrations can use the following POS Snippets:

- **/QuickSignup** – Provides simple way for a cashier to lookup and/or create a new customer in a single call.
- **/FindCustomer** – Presents the cashier with the bLoyal find customer screen where the cashier can easily find the customer, scan a loyalty card, and create new customers that don't yet exist.
- **/CheckBalance** – Checks the balance of a Gift Card, eGift Card, Loyalty Card, or Certificate award (aka Voucher).
- **/ViewCustomer** – Provides a full view of the customer to enable the cashier to better serve the customer. Includes current loyalty balances, purchase history, program and club signup, pickup orders, etc…
- **/OfferDetails** – Provides more detail on offers returned by the CalculateSalesTransaction() method. This snippet takes an "OfferToken=####" on the query string.
- **/ApplyCoupon** – A window for the cashier to scan in a coupon or lookup the awarded coupons for a customer and apply a coupon to a transaction.
- **/RecordVisit** – Records a visit for a customer. If no customer has been set to the transaction then it provides UI to scan or enter customer information.

Order Engine related POS Snippets:

- **/CreateOrder** – Creates either a shipping order or pickup order from the current POS transaction. You will need to call CalculateSalesTransaction() before calling the CreateOrder snippet. This is how the products on the transaction are made available to the CreateOrder snippet. You should also expect to get shipping charges and destination based sales taxes back as line items from CalculateSalesTransaction the next time it is called if the cashier converted the transaction to an order.

- **/PickupOrder** – The Pickup order snippet will show a list of pick orders for the customer and enable the cashier to mark an order as picked up by the customer. The pickup orders are also available as a section in the ViewCustomer snippet.
- **/ReturnOrder** – Any order or POS transaction that bLoyal know about can be returned using the ReturnOrder POS snippet. If a return is requested by the cashier then it will be returned the next time the CalculateSalesTransaction method is called.
- **/CheckInventory** – This snippet will show the inventory that is available for a given product across the entire organization. This snippet takes an additional LookupCode query string parameter for the product to return inventory for. If no LookupCode is passed in then the snippet will enable the cashier to enter a lookup code manually.

6. Channel Hub™ – Backend Data Integration

The bLoyal Channel Hub™ is a highly scalable integration service that synchronizes key data between bLoyal and 3rd party applications. The Channel Hub takes care of most of the heavy lifting required to synchronize data between systems including dynamic identification and duplicate resolution using bLoyal's patent pending Dynamic Resolution Technology™ and merge conflicts and exceptions.

The Channel Hub is exposed through the ChannelHub.svc web service:

https://ws.bloyal.com/v3.5/ChannelHub.svc

6.1 Protocol – Register() and Checkin()

Unlike the LoyaltyEngine.svc web service where methods are called in real time (synchronously), the Channel Hub web service is intended to be called as part of a background process within the POS system.

The protocol for synchronizing data using the Channel Hub web service involves calling a Register() method once and then calling a Checkin() method on a periodic basis.

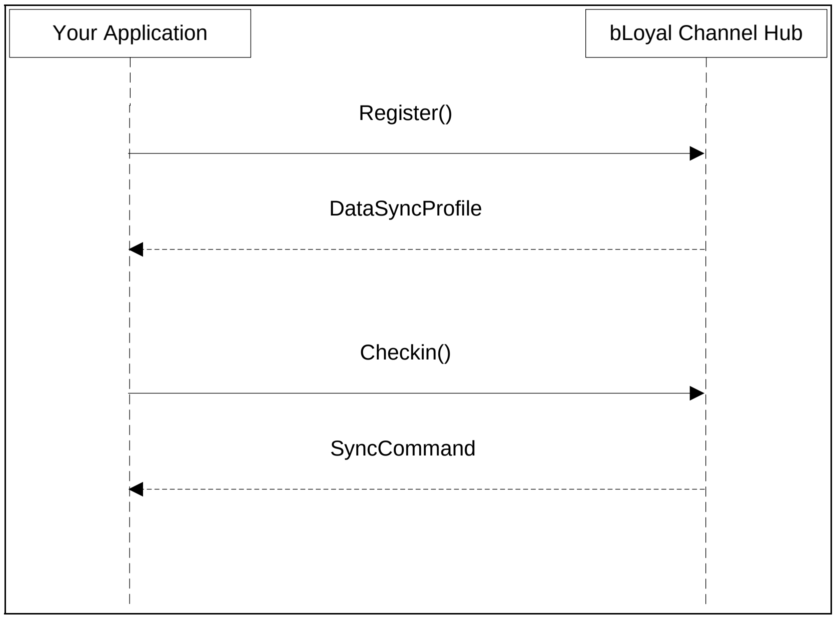

Figure 1 – Channel Hub Protocol

The Register() and Checkin() methods are described below:

- **Register()** – The Register method is made once when your service starts up or when commanded by the Checkin() call below. The Register method returns a secure session key and a data synchronization profile that tells your application which data needs to be synchronized with the Channel Hub. A bLoyal client defines a data synchronization profile in their Loyalty Director account that specifies the data they want synchronized between systems. This profile is returned by the Register method. Additionally the Register method will return the following statuses:

  Normal – SyncProfile

Exception:

- **Redirect Exception**
- **LockedOut Exception –** This tells your replication service that it has been locked out for some reason. For example, this would happen if the bLoyal Client decided to no longer replicate data with the system or otherwise temporarily disabled the replication for some reason.

- **Checkin()** – the Checkin method should be called on a periodic basis, usually once per minute, and returns one of the following SyncCommand values that tell your application what it should do.

  The following are SyncCommands returned as part of normal processing.
  - **SyncCommand.Synchronize –** This tells your replication service to synchronize data. This is the normal SyncCommand value returned by the Checkin method.
  - **SyncCommand.NoAction –** This tells your replication service to not do anything and provides a delay for when to call Checkin next. This would normally be a 1 minute delay.
  - **SyncCommand.Register –** This tells your replication service to reregister (call the Register method again). This is an infrequent command that would be returned if the bLoyal Client changed their synchronization profile or the session key needs to be renewed.

  The following SyncCommands can also be returned for unusual events.
  - **SyncCommand.Busy –** This tells your replication service that the Channel Hub is busy and will provide a delay for when to check in next. This would happen infrequently and would normally be based on a system update.

6.2 Synchronizing Entities

When the Checkin() method returns the SyncCommand.Synchronize command your application can then replicate each of the entities specified in the DataSyncProfile returned by the Regsiter() method. Your system should send all the changes that have happened since the last Checkin and then immediately call the Checkin method again to tell the Channel Hub that you are done with the current change set.

There are four method calls you will need to make for reach entity to the synchronized. The order in which you make the calls is very important. For a given entity you should first let the Channel Hub service know of any changes to the entity and then get the changes that the Channel Hub has for your system.

See the sequence diagram below.

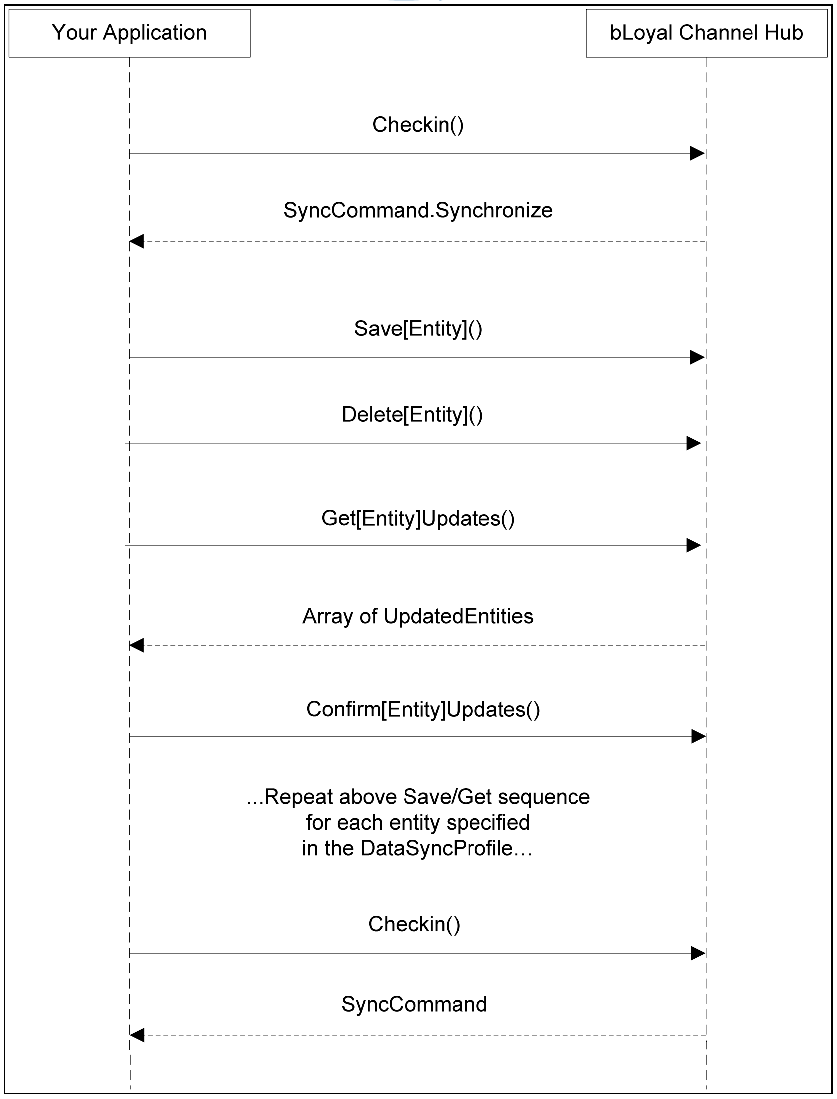

Figure 2 – Entity Sync

There are four method calls you will need to make for reach entity to the synchronized. ***Note:*** The [Entity] token in the name can be replaced by the name of the specific entity to be synchronized (e.g. SaveCustomer()).

- **Save[Entity]()** – Your system needs to detect and keep track of when changes occur in your system. This is normally done either by setting a "HasChanged" flag on the entity or keeping a queue/table of ID's for the entities that have changed. You should call the Save method for each entity that has changed since the last time you sent changes.

- **Delete[Entity]()** – The Delete method is called when an entity has been deleted from your system.

- **Get[Entity]Updates()** – The Get[Entity]Updates method returns a list of entities that have been added, updated, or deleted in bLoyal. You will get up to 100 updates in a single call. ***Important:*** If you receive an entity from bLoyal that has been changed locally in your system (since the last Save[Entity] call), then you should reject the change and the bLoyal resolution and exception logic will take of the issue on the next Checkin call.

After processing the updates you need to call the Confirm[Entity]Updates method. You should then call Get[Entity]Updates until no more updates are available for you to process.

- **Confirm[Entity]Updates()** – The Confirm[Entity]Updates method tells the Channel Hub that you have processed each of the updates and have either accepted or rejected the change and are ready for the next list of updates.

6.3 Supported Entities

The bLoyal Channel Hub supports replicating the following entities between systems.

Entities used by the bLoyal Promotions and Loyalty Engine:

- **Customers**
- **Products** – Core product data such as SKU, name, price, etc…
- **Departments** – A top level grouping of a product normally in a POS system.
- **Categories** – A category is normally below a department
- **SalesTransactions** – Sales Transactions can be replicated via a backend process using the Channel Hub or directly in real time using the LoyaltyEngine.svc web service directly.
- **ReasonCodes** – Reason codes are used on discounts and other events within bLoyal. You can choose to replicate reason codes between your system and bLoyal.

Additional entities used by the Order Engine:

- **ProductInventory** – The ProductInventory entity represents the current inventory quantity (on-hand, committed, and offline) for a given product at an inventory location. bLoyal tracks inventory but is not the inventory management system so this number is a snapshot in time from your system. It effectively resets the inventory that bLoyal has for a product.
- **InventoryTransactions** – An inventory transaction is generated by bLoyal each time an order is approved (committed inventory) or fulfilled (onhand inventory). Inventory transactions are sent to your system for each inventory location you're your system is managing.
- **ClubMemberships** – bLoyal provides club members signup and maintenance, and club order processing. However, Club Membership data can be replicated to a 3rd party application for cases when bLoyal is handling the club signup and member management but not the club order processing system.
- **ProductCatalogs** – bLoyal provides a flexible product catalog capability for grouping products based on an unlimited number of criteria and levels. bLoyal Clients often use this for the classification for their web store for example.

7. Revision History

Date	Rev	Person	Description
1/4/2014	1	Robert Carney	Created for the v3.5 web services and the 2014-R1 bLoyal release.
2/27/2014	2	Robert Carney	Updated the document to reflect pre-release changes to the LoyaltyEngine.svc web service and the POS Snippet parameters.